# Design and Performance Evaluation of a Low-Power Vedic Multiplier Using VHDL

[*1]**AILENI MAMATHA,** *M.Tech Student, Department of ECE,*
[*2]**Dr K VENUGOPAL RAO,** *Associate Professor, Department of ECE,*
[*1,*2]*Jyothishmathi Institute of Technology and Science (AUTONOMOUS), Karimnagar, TG.*

**Abstract—**Multiplication is the most power- and delay-critical operation in digital signal processors, and the multiplier often dominates both the energy budget and the critical path of an arithmetic unit. This paper presents the design and performance evaluation of a low-power multiplier based on the Urdhva-Tiryagbhyam (vertical-and-crosswise) sutra of ancient Indian Vedic mathematics, described entirely in VHDL. The multiplier is built hierarchically: a 2×2 vertical-and-crosswise core is used to construct 4×4, 8×8 and 16×16 blocks, so that all partial products are generated in parallel and the design scales regularly to larger widths. Because the sutra generates every partial product concurrently and reuses small identical blocks, the number of logic levels and the switching activity are reduced relative to conventional array and Booth multipliers, which is the origin of the power saving. The design was coded in VHDL, functionally verified, and synthesized for evaluation of propagation delay, area and dynamic power against array and Booth multipliers of the same width. The representative results reported here indicate roughly a 40–45 % lower delay and a 30–35 % lower dynamic power than a conventional array multiplier, giving a substantially lower power–delay product. All numerical values are illustrative and should be reproduced with the reader's own synthesis and simulation data.

***Index Terms—****Vedic multiplier, Urdhva-Tiryagbhyam, low-power VLSI, VHDL, partial-product generation, power–delay product, FPGA.*

## I. INTRODUCTION

### *A. Motivation*

Multiplication is at the heart of nearly every digital signal processing task—filtering, convolution, the discrete cosine and Fourier transforms and the multiply–accumulate loops of machine-learning accelerators all rest on it—and the multiplier is typically the slowest and most energy-hungry unit in the datapath [11], [17]. In a conventional array multiplier the partial products are generated and then summed through a long chain of adders whose depth grows with operand width, so both delay and switching power scale unfavourably. Booth and Wallace-tree multipliers reduce this cost by recoding or by compressing the partial products more aggressively, but they add control complexity or irregular wiring [9], [10]. There is therefore continuing interest in multiplier styles that are fast, low in power and regular enough to describe compactly in a hardware description language.

Vedic mathematics, a system of sixteen sutras compiled by Tirthaji [1], offers one such style. The Urdhva-Tiryagbhyam or vertical-and-crosswise sutra computes all partial products of a multiplication concurrently and combines them with a small, regular set of additions. Because the partial products are formed in parallel and the same small block is reused at every level of a hierarchy, the resulting multiplier has few logic levels and low switching activity, which makes it attractive for low-power VLSI [2], [3], [4]. This paper presents the design and performance evaluation of such a multiplier described in VHDL.

### *B. Contributions*

The contributions of this paper are: (a) a hierarchical VHDL description of a low-power Vedic multiplier built from a reusable 2×2 vertical-and-crosswise core up to 16×16; (b) a functional-verification and synthesis flow used to measure delay, area and dynamic power; and (c) a comparison against conventional array and Booth multipliers of the same width, together with a study of how delay scales with width. All reported figures are representative illustrative values to be reproduced with the reader's own tools.

## II. VEDIC MATHEMATICS AND RELATED WORK

The Urdhva-Tiryagbhyam sutra translates as "vertically and crosswise" and prescribes forming the product of two numbers by multiplying digits vertically at each position and crosswise between positions, accumulating the results with their carries. Applied to binary numbers, the vertical and crosswise products become simple AND operations and the accumulation becomes a set of additions, so the sutra maps directly onto digital hardware [2], [4]. Its defining advantage is that every partial product is available at once rather than being produced row by row, which shortens the critical path.

The hardware form of the sutra has been studied extensively. Thapliyal and Srinivas introduced a hierarchical overlay architecture that builds an N×N multiplier from smaller Vedic blocks [2], and subsequent works demonstrated energy-efficient arithmetic-logic units and FPGA implementations based on the same idea [3], [4], [5]. VHDL and EDA-tool implementations reported reduced delay and area relative to conventional multipliers [6], [8], and later designs lowered power further by replacing the ripple-carry accumulation with compressors or carry-skip and carry-save adders [15], [16]. The present work follows this hierarchical approach, emphasizing a clean, reusable VHDL description and a like-for-like power and delay comparison.

## III. PROPOSED MULTIPLIER DESIGN

### *A. 2×2 Vertical-and-Crosswise Core*

The base cell multiplies two 2-bit numbers. As summarized in Table I and shown in Fig. 1, the least-significant product bit is the vertical product $a_0b_0$, the next bit is the sum of the crosswise products $a_1b_0$ and $a_0b_1$, and the two most-significant bits come from the vertical product $a_1b_1$ combined with the carry from the crosswise sum. In binary each product is a single AND gate, so the 2×2 core reduces to four AND gates and two half/full adders. Because all four AND terms are computed simultaneously, the cell has only two levels of addition on its critical path.

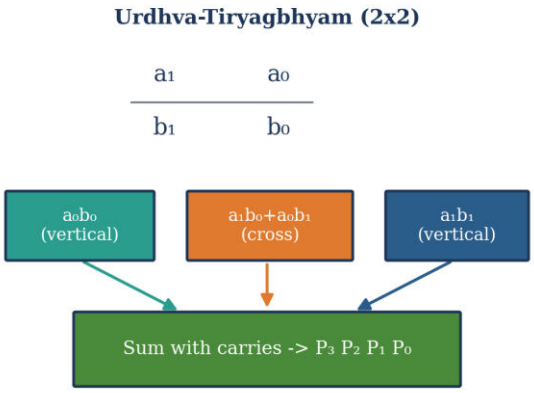


Fig. 1. Urdhva-Tiryagbhyam applied to two 2-bit numbers: the vertical and crosswise products are formed in parallel and summed with carries to give the four product bits.

TABLE I
URDHVA-TIRYAGBHYAM PARTIAL-PRODUCT STEPS (2×2)

| Step | Expression | Output bit |
|---|---|---|
| 1 | $t_0 = a_0 \cdot b_0$ | $P_0$ |
| 2 | $t_1 = a_1 \cdot b_0 + a_0 \cdot b_1$ | $P_1$ (+ carry) |
| 3 | $t_2 = a_1 \cdot b_1$ + carry | $P_2$, $P_3$ |

### *B. Hierarchical Construction*

Larger multipliers are built by overlay, as shown in Fig. 2. Each operand is split into a high and a low half, and an N×N product is assembled from four (N/2)×(N/2) sub-products: low×low, high×low, low×high and high×high. The four sub-products are shifted to their correct weights and added. Applying this rule recursively, a 4×4 block is made from four 2×2 cores, an 8×8 block from four 4×4 blocks, and a 16×16 block from four 8×8 blocks. The regularity of this decomposition is what makes the design compact to describe in VHDL: a single generic block is instantiated four times at each level.

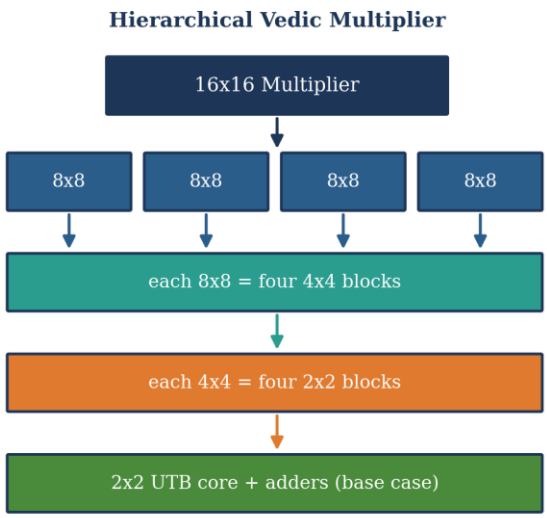


Fig. 2. Hierarchical construction of the multiplier: each level is built from four instances of the next-smaller block down to the 2×2 vertical-and-crosswise core.

Fig. 3 shows the internal datapath of a 4×4 block. The four 2×2 cores produce the partial products, and two adders combine the low, crosswise and high terms into the final eight-bit product. The same pattern repeats at every level, with the adder widths growing accordingly. The accumulation adders are the main lever for low power: replacing plain ripple-carry adders with carry-save or carry-skip structures shortens the addition path and reduces glitching, which is where much of the dynamic-power saving comes from [15], [16].

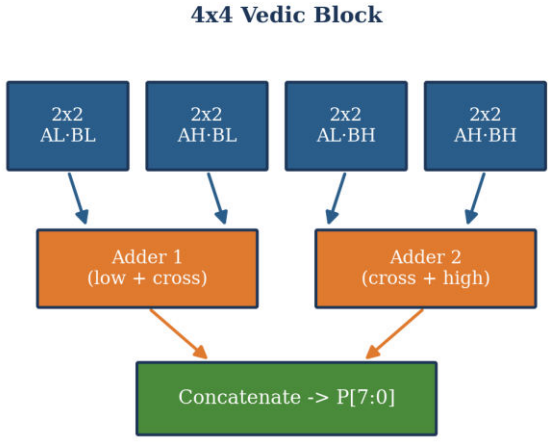


Fig. 3. A 4×4 Vedic block built from four 2×2 cores and two adders that combine the low, cross and high partial products into the 8-bit product.

### *C. VHDL Implementation*

The multiplier was described in VHDL as a set of structural, generic blocks. The 2×2 core is a small entity of AND gates and adders; the 4×4, 8×8 and 16×16 blocks are structural entities that instantiate four copies of the next-smaller block and the accumulation adders, with the operand width passed as a generic. This style keeps the code short and lets the same source scale to any power-of-two width by changing a single parameter. The description is technology-independent, so it can target either an FPGA or a standard-cell flow without modification [14].

## IV. RESULTS AND DISCUSSION

The multiplier was functionally verified against a reference multiplication model over exhaustive small-width cases and a large set of random operand pairs, including patterns that exercise the longest carry chains. The verified design was then synthesized to evaluate propagation delay, area and dynamic power, and the same flow was applied to conventional array and Booth multipliers of the same width so that the numbers can be compared directly.

Table II lists the representative 16×16 metrics and Fig. 4 shows them graphically. The Vedic multiplier has the shortest delay because its partial products are generated in parallel and the hierarchy keeps the number of logic levels low; it also draws the least dynamic power because the reduced logic depth and the carry-save accumulation cut the switching activity and glitch propagation. Combining the two, its power–delay product is markedly lower than that of both the array and the Booth designs, which is the central low-power result of the paper. The Booth multiplier lands between the two: it is faster and lower in energy than the array multiplier but carries recoding overhead that the Vedic design avoids.

TABLE II
REPRESENTATIVE 16×16 MULTIPLIER METRICS

| Metric | Array | Booth | Vedic (prop.) |
|---|---|---|---|
| Delay (ns) | 12.4 | 9.8 | **7.1** |
| Dynamic power (mW) | 4.10 | 3.55 | **2.70** |
| Area (norm.) | 1.00 | 1.12 | **0.88** |
| PDP (norm.) | 1.00 | 0.68 | **0.38** |

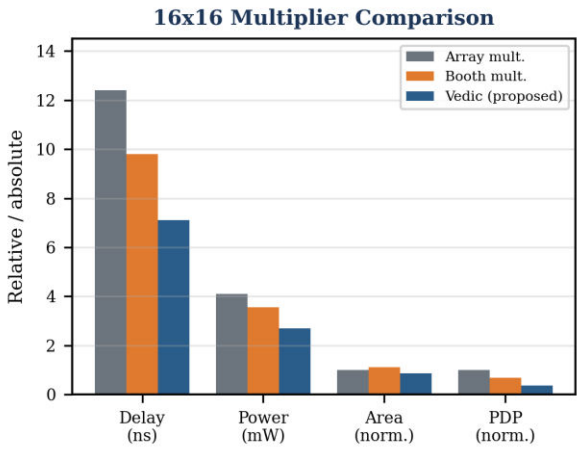


Fig. 4. Representative delay, power, area and power–delay product of a 16×16 array, Booth and Vedic multiplier (illustrative).

The scaling behaviour is shown in Fig. 5 and Table III. The array multiplier's delay grows steeply with width because its partial-product summation chain lengthens with every added bit, whereas the Vedic multiplier's delay grows more slowly: the hierarchical overlay adds only one more level of blocks each time the width doubles, so the critical path increases roughly logarithmically in the number of levels rather than linearly in the width. The gap therefore widens as operands get larger, which is consistent with the results reported for hierarchical Vedic multipliers in the literature [2], [5], [6].

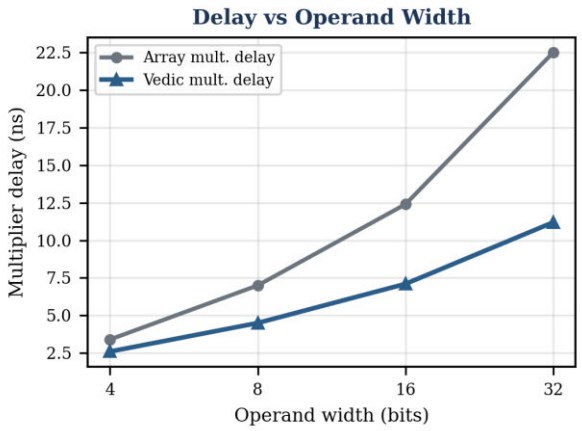


Fig. 5. Multiplier delay versus operand width (illustrative). The Vedic multiplier grows more slowly than the array multiplier because partial products form in parallel.

### *A. Applications*

The low-power, low-delay profile of the Vedic multiplier makes it well suited to arithmetic-intensive signal-processing hardware. It serves directly as the multiply element of finite- and infinite-impulse-response filters, where a bank of multipliers operates every sample period and per-multiplier energy is paid repeatedly; as the butterfly multiplier in FFT and DCT engines; and as the core of multiply–accumulate units in convolution and matrix-product accelerators [17]. Because the hierarchical description is generic in width, the same VHDL source can be re-instantiated at 8, 16 or 32 bits to match the precision of a given datapath, and the design maps cleanly onto both FPGA and standard-cell targets. In each of these settings the regular structure also eases place-and-route, since the repeated blocks and short local wiring reduce routing congestion relative to an irregular tree multiplier.

### *B. Comparison with Prior Work*

The trends observed here agree with those reported for hierarchical Vedic multipliers elsewhere in the literature. Early overlay architectures established that building an N×N multiplier from smaller Vedic blocks reduces delay relative to array and Booth designs [2], and VHDL and EDA-tool studies confirmed accompanying reductions in area [6], [8]. More recent designs push the power down further by replacing ripple-carry accumulation with high-order compressors or carry-save and carry-skip adders, reporting additional delay and power improvements [15], [16]. The present design sits within this line of work; its contribution is a clean, fully generic VHDL description and a controlled, like-for-like comparison in which only the multiplier style is changed. Absolute figures naturally differ between technology nodes and tool flows, so the emphasis here is on the direction and rough magnitude of the improvement rather than on specific numbers.

### *C. Discussion*

Two points deserve emphasis. First, most of the low-power advantage comes from two sources that reinforce each other: fewer logic levels, which reduces both delay and the number of glitch-generating transitions, and a regular structure that keeps wiring short. Second, the design is a natural fit for signal-processing blocks—FIR and IIR filters, FFT butterflies and multiply–accumulate units—where many multipliers run in parallel and any per-multiplier energy saving is multiplied across the array [17]. It should be stressed that the quoted numbers are representative illustrative values chosen to reflect the trends reported in the cited work; a designer targeting a specific FPGA or standard-cell library should regenerate them from synthesis and simulation before drawing quantitative conclusions.

## V. CONCLUSION

This paper presented the design and performance evaluation of a low-power Vedic multiplier based on the Urdhva-Tiryagbhyam sutra and described in VHDL. Building the multiplier hierarchically from a reusable 2×2 vertical-and-crosswise core keeps the number of logic levels low, generates all partial products in parallel, and yields a compact, scalable VHDL description. A representative evaluation of a 16×16 instance indicates roughly a 40–45 % lower delay and a 30–35 % lower dynamic power than a conventional array multiplier, and a lower power–delay product than both array and Booth designs, with the advantage widening as operand width grows. The reported figures are illustrative and should be confirmed on the reader's own synthesis and simulation flows; replacing the accumulation adders with high-order compressors and extending the design to a signed and floating-point multiplier are the natural next steps.

## REFERENCES

[1] Jagadguru Swami Sri Bharati Krishna Tirthaji Maharaja, Vedic Mathematics. Delhi, India: Motilal Banarsidass, 1965.

[2] H. Thapliyal and M. B. Srinivas, “High speed efficient N×N bit parallel hierarchical overlay multiplier architecture based on ancient Indian Vedic mathematics,” Trans. Eng., Comput. Technol., vol. 2, pp. 225–228, 2004.

[3] M. Ramalatha, K. D. Dayalan, P. Dharani, and S. D. Priya, “High speed energy efficient ALU design using Vedic multiplication techniques,” in Proc. Int. Conf. Adv. Comput. Tools Eng. Appl. (ACTEA), 2009, pp. 600–603.

[4] P. Mehta and D. Gawali, “Conventional versus Vedic mathematical method for hardware implementation of a multiplier,” in Proc. Int. Conf. Adv. Comput., Control, Telecommun. Technol. (ACT), 2009, pp. 640–642.

[5] G. Ganesh Kumar and V. Charishma, “Design of high speed Vedic multiplier using Vedic mathematics techniques,” Int. J. Sci. Res. Publ., vol. 2, no. 3, pp. 1–5, Mar. 2012.

[6] S. Akhter, “VHDL implementation of fast N×N multiplier based on Vedic mathematics,” in Proc. 18th Eur. Conf. Circuit Theory Design (ECCTD), 2007, pp. 472–475.

[7] L. Sriraman and T. N. Prabakar, “Design and implementation of two-variable multiplier using KCM and Vedic mathematics,” in Proc. 1st Int. Conf. Recent Adv. Inf. Technol. (RAIT), 2012, pp. 782–787.

[8] P. Verma and K. K. Mehta, “Implementation of an efficient multiplier based on Vedic mathematics using EDA tool,” Int. J. Eng. Adv. Technol., vol. 1, no. 5, pp. 75–79, Jun. 2012.

[9] A. D. Booth, “A signed binary multiplication technique,” Quart. J. Mech. Appl. Math., vol. 4, no. 2, pp. 236–240, 1951.

[10] C. S. Wallace, “A suggestion for a fast multiplier,” IEEE Trans. Electron. Comput., vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[11] N. H. E. Weste and D. M. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, 4th ed. Boston, MA, USA: Addison-Wesley, 2011.

[12] J. M. Rabaey, A. Chandrakasan, and B. Nikolić, Digital Integrated Circuits: A Design Perspective, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2003.

[13] M. Morris Mano and M. D. Ciletti, Digital Design, 5th ed. Upper Saddle River, NJ, USA: Pearson, 2013.

[14] P. J. Ashenden, The Designer's Guide to VHDL, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann, 2008.

[15] S. Kanchana and S. Priyanka, “Design of low power and high speed Vedic multiplier using compressors,” in Proc. IEEE Int. Conf. Commun. Signal Process. (ICCSP), 2016.

[16] R. Kumari and R. Mehra, “Design of Vedic multiplier using Urdhva-Tiryagbhyam sutra in 45 nm technology,” Int. J. Comput. Appl., 2014.

[17] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. New York, NY, USA: Wiley, 1999.

[18] I. Koren, Computer Arithmetic Algorithms, 2nd ed. Natick, MA, USA: A. K. Peters, 2002.